Tools for Working with Data

Nicole Sorhagen, Ph.D.

2020-06-02

Contents

Chapter 1

About this book

This book describes how to use R statistics as a tools for working with data.

R statistics is becoming increasingly popular for data management and analysis due to its accessibility and versatility. For example, R can produce records of data analyses, which is consistent with the growing move towards reproducible and open science within the field of psychology. R statistics is also known for making elegant graphs, which can help develop students' data visualization skills. Because it is open-sourced it is extremely flexible - people create and share packages that make certain aspects of data analysis easy.

R is a programming language. Although learning a programming language can seem a bit intimidating, there are many benefits to trying to figure it out. Mastering the basics of R could be useful for your future coursework, as well as for data management and analysis needs outside the classroom (independent research, future employment, etc.). Learning the basics of a programming language is a highly transferable skill.

R is free so you can have it on your own computers - eliminating the need to visit computer labs or to buy student versions of expensive software. R is a programming language that can be used within the R software as well as other programs. R can be downloaded from the CRAN (Comprehensive R Archive Network) (https://www.r-project.org/). Rstudio is a free IDE (integrated development environment) for the R programming language (https://rstudio.com/). Rstudio cannot run without R (R can be used without Rstudio).

While you are welcome to download R and Rstudio on your personal computer, you do not have to for this course. We will be using Rstudio in the cloud for class work.

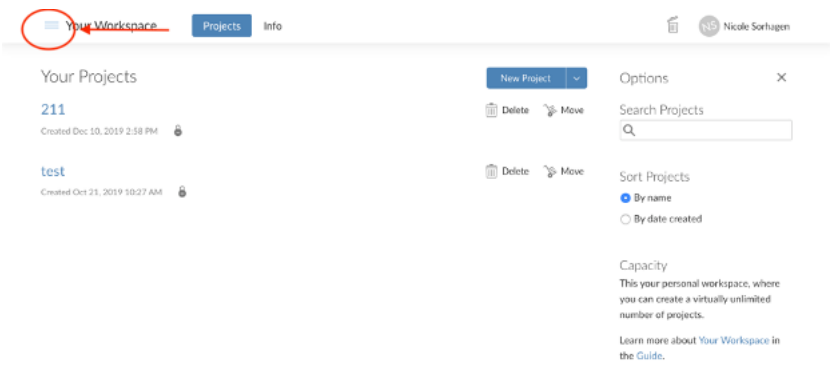
Finally, please note that I will be updating this book over the course of the semester.

This work is licensed under a Creative Commons Attribution-NoDerivatives 4.0 International License.

Chapter 2

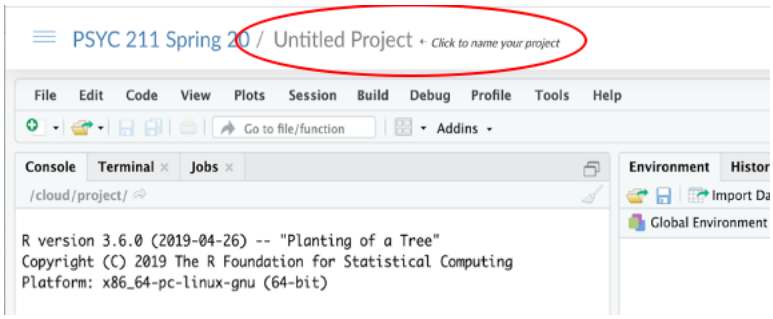
Set up project on Rstudio Cloud

First expand the R studio cloud options by clicking on the 3 lines in the top left corner.



Then select PSYC 211 for the current semester. If you cannot see this option – then you have not been added to our shared workspace. See the Introduction to R – overview (start here) html site on this week's D2L site for instructions on how to get into our shared workspace.

Once you are in the shared PSYC 211 workspace, open a new project and title it with your last name by clicking on the box that says 'Untitled Project' and typing your last name.



I will be able to see everyone's project. But you will only be able to see your project and my project.

Chapter 3

Introduction

This chapter introduces the Rstudio cloud environment and describes how to import data into the RStudio cloud.

R cannot handle typos and is case sensitive ('Gender' is not the same as 'gender'). If your code will not run check for typos and caps.

3.1 Layout of Rstudio cloud

Rstudio has four panes: the console panel, the script panel, the environment and history panel, and the files and plots panel. Each will be describe in turn next.

3.1.1 Console

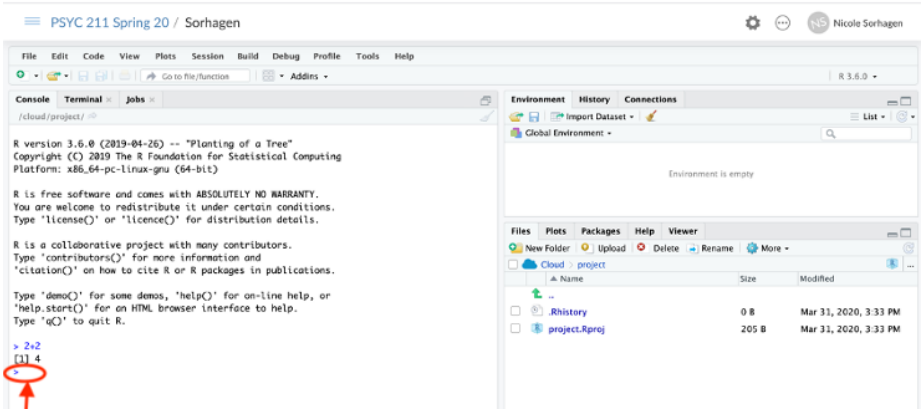
The console panel of R studio is where you can type commands and where you will see the output of commands.

In its most basic form, you can think of R as a fancy calculator.

For example:

In the console type `2+2` and then press RETURN on your keyboard. The answer '4' will apear on the next line.

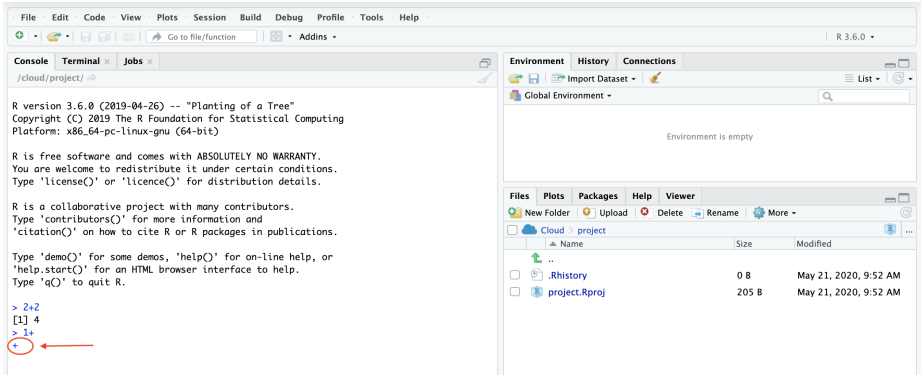
The `>` in the last line of the console means that the console is ready for a command (see red circle in the picture above).



If > is missing from the last line, it means that R is waiting for you to complete a command.

For example, type `1+` in the console and then hit enter.

The plus sign means the command is incomplete.



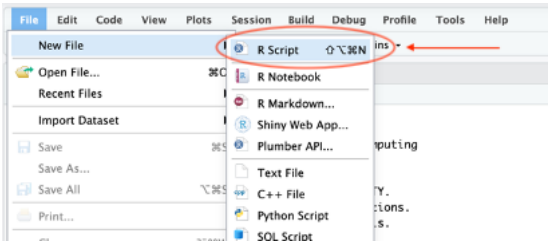
Push the **ESC** button on your keyboard to get back to the command prompt.

3.1.2 Script

One of the benefits of using R is that you can save a record of your work using scripts. Records of your work allow you to easily start and stop an assignment or research project. You can pick up where you left off whether it is 20 minutes later or 2 years later. It also lets you share with others – from professors, to collaborators, to peer reviewers.

To create a new script, go to the top bar menu:

FILE -> NEW FILE -> R SCRIPT



A new script will open in the top left of the RStudio platform.

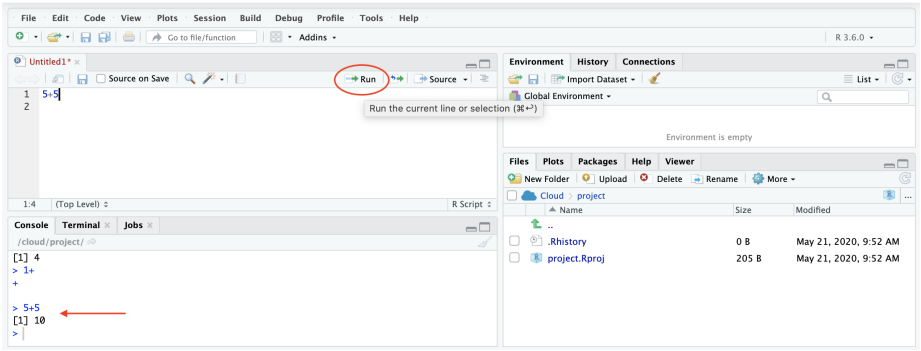
You should run code from scripts

Scripts are similar to running command in the console (this is what you did in the last section).

For example, type `5+5` in the script panel.

In order to run command in a script you should click the run button while the cursor is in the code or the code is selected. You can also run the code by pressing the COMMAND and RETURN keys on your keyboard at the same time (the ALT and RETURN key on a pc).

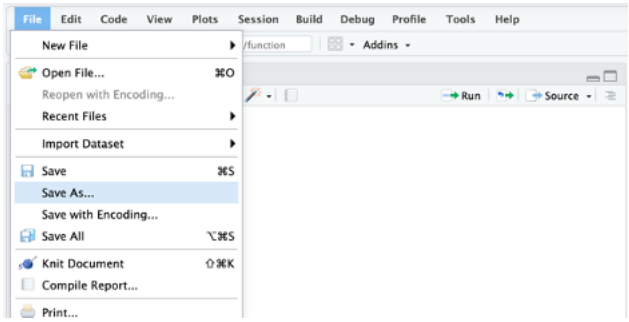
After the code is run, the results will automatically appear the in console (see red arrow in the picture below).



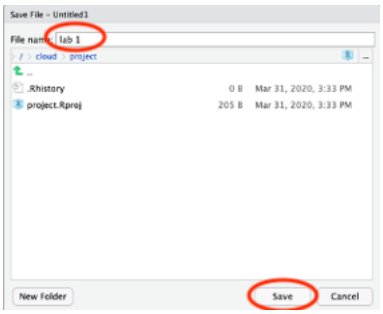
In order to use the script again you must **save** it.

From the drop-down menu select:

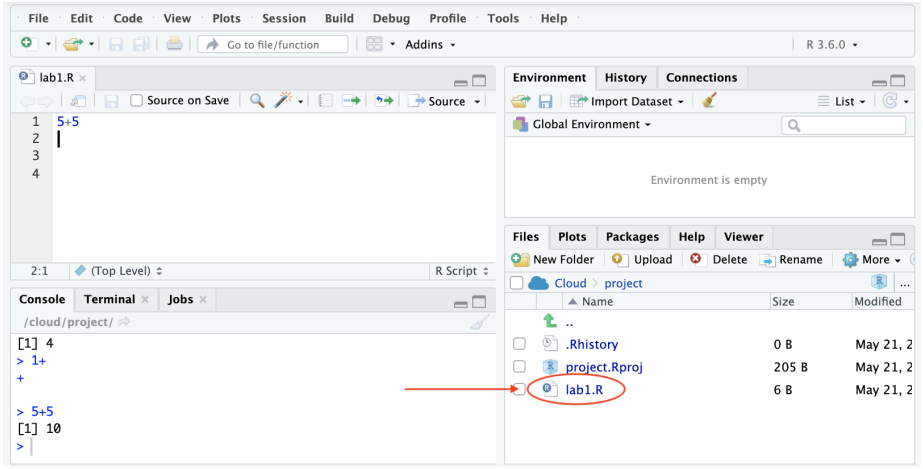
FILE -> SAVE AS



Type **lab 1** into the file name box. And then click the SAVE button.



Your file should now be listed in the files window in the bottom right.



This script file is a record of your work and is how you will be graded for this lab. Make sure you saved this file and complete the rest of this lab in your 'lab1' script.

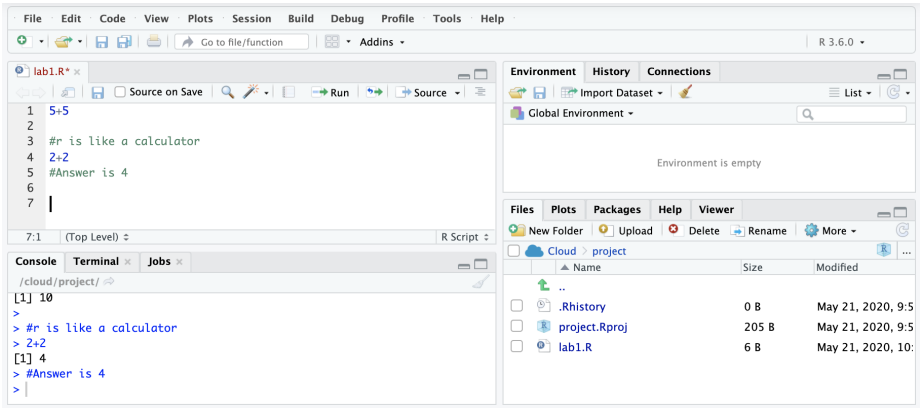
Within a script you should include comments to yourself and others using **#**. Anything with a **#** in front of it will not run. These comments and explanations are an important part of an R script.

For example, type the following in to the script and then run it.

```
#r is like a calculator

2+2

#Answer is 4
```



Note that the comments are green in the script.

3.1.3 Environment and history

In the top right corner of RStudio is the environment and history window. The **history tab** shows every line of code that has been run in the current session.

The **environment tab** is where all active **objects** are listed. An object is something can hold information for later use. The information can be data, values, output, or functions.

Objects are assigned using `<-`. Values on the right side of `<-` will be assigned to the object on the left side.

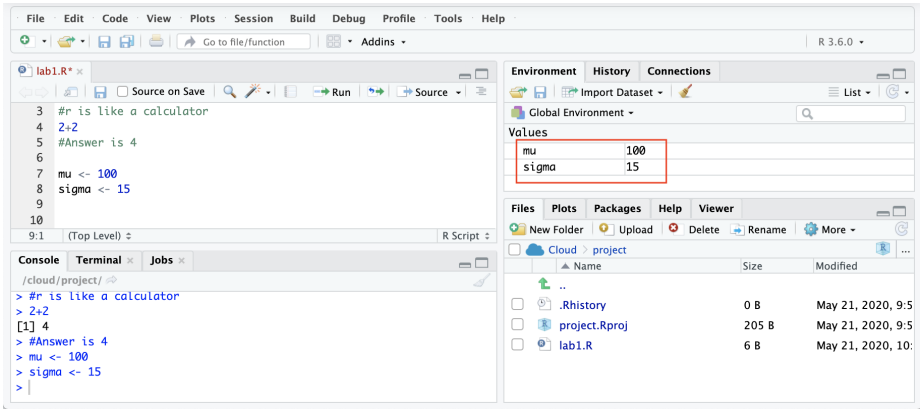
For example, let's tell R that the population mean of IQ scores is 100 and the population standard deviation is 15.

To do this use the following code:

```
mu <- 100

sigma <- 15
```

After you run these commands, the objects will now be listed in the environment panel in the top left.



The shortcut for making `<-` is the ALT and – key together. (or OPTION and – on a mac)

3.1.3.1 Vectors

It is possible to store more than one number in an object. One way to do this is to use a a **vector**. Assign a set of numbers a vector with the **combine** function: `c()`. Do use this, type all the numbers you want to store within the parentheses in a comma separated list.

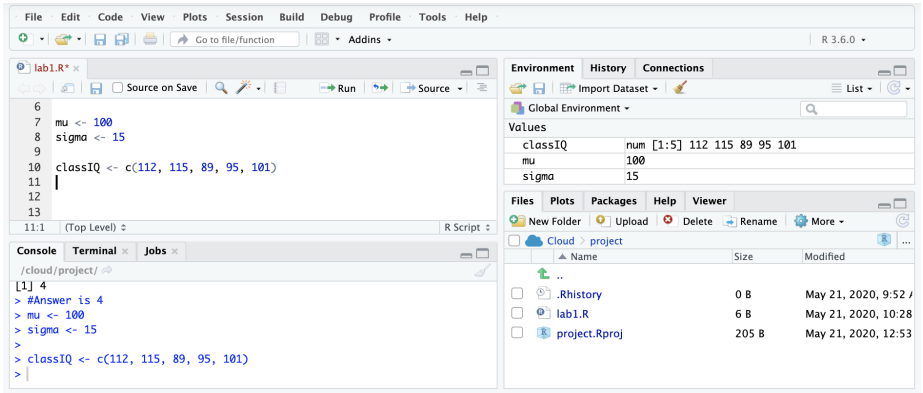
For example, let's enter IQ scores of students in a small class.

To do this use the following code:

```
classIQ <- c(112, 115, 89, 95, 101)
```

After you run this code,the classIQ vector should appear in the environment.

Here is a picture of what your screen should look like:



Calculations with vectors apply to all data points.

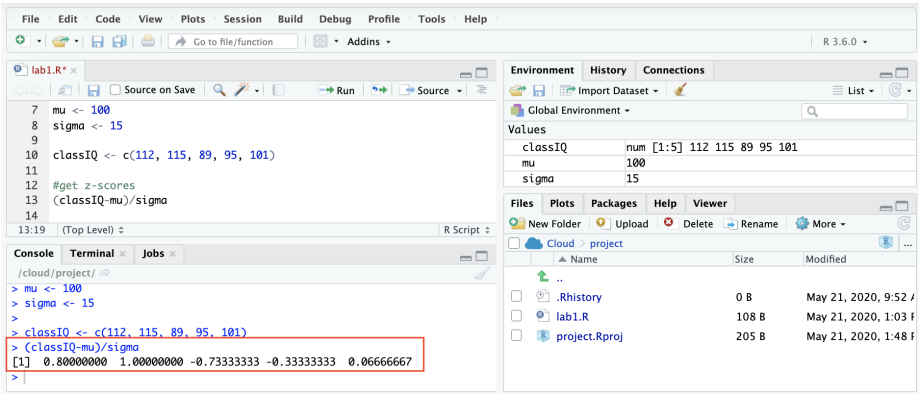
For example, let's calculate the z-scores for each of the IQ scores.

To do this use the following code:

```
#get z-scores

(classIQ-mu)/sigma
```

The results will appear in the console (See the red box in the picture below)



It is possible to save these answers as a vector using the `<-` function.

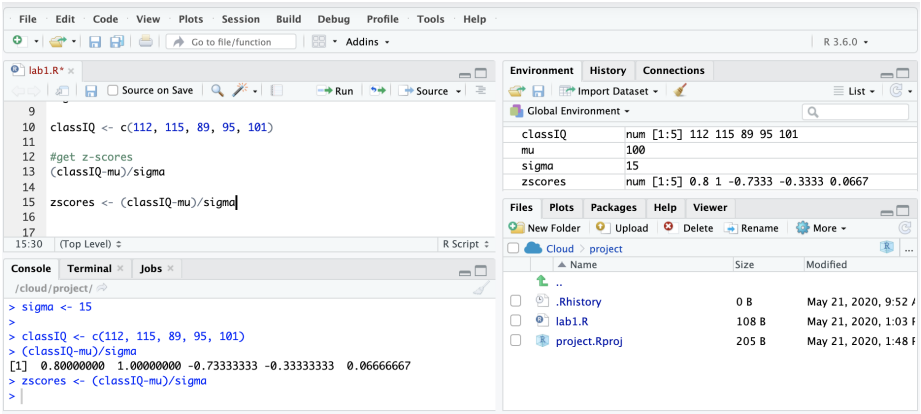
For example, let's save those zscores in a vector called zscores.

To do this use the following code:

```
zscores <- (classIQ-mu)/sigma
```

There should now be a vector in the environmnet called zscores.

Here is a picture so that you can check your progress:



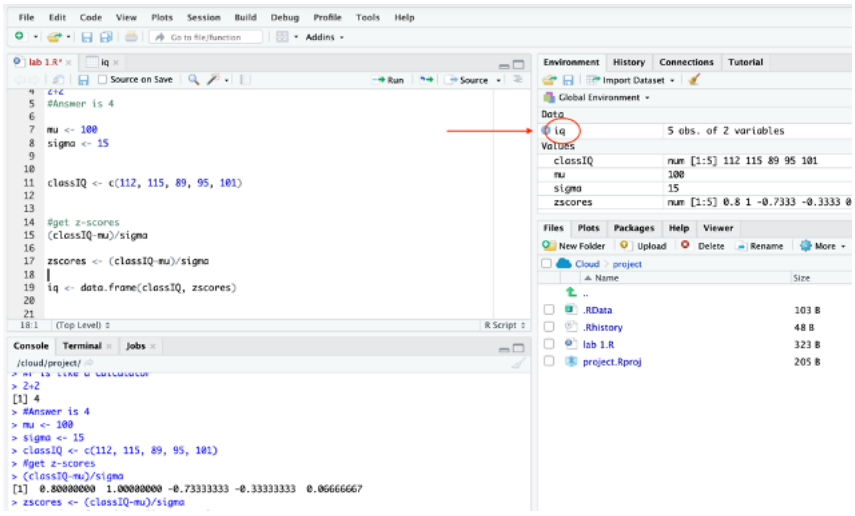
3.1.3.2 Data frames

Right now the IQ scores and the z-scores are in separate objects. Variables often need to be in a single object in order to do some basic analyses. You can combine the classIQ and the zscores variable using the **data.frame** command.

To do this use the following code:

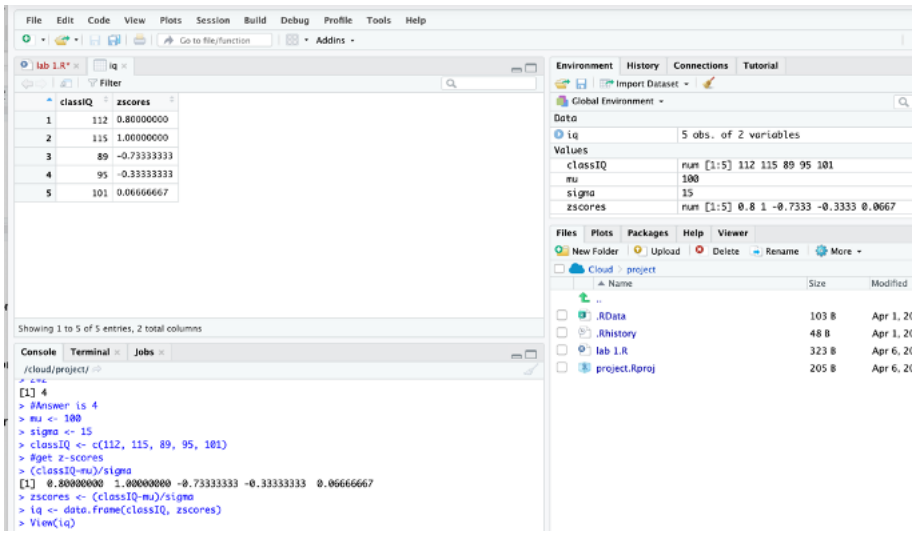
```
iq <- data.frame(classIQ, zscores)
```

This object will be listed under data instead of values in the environment panel.



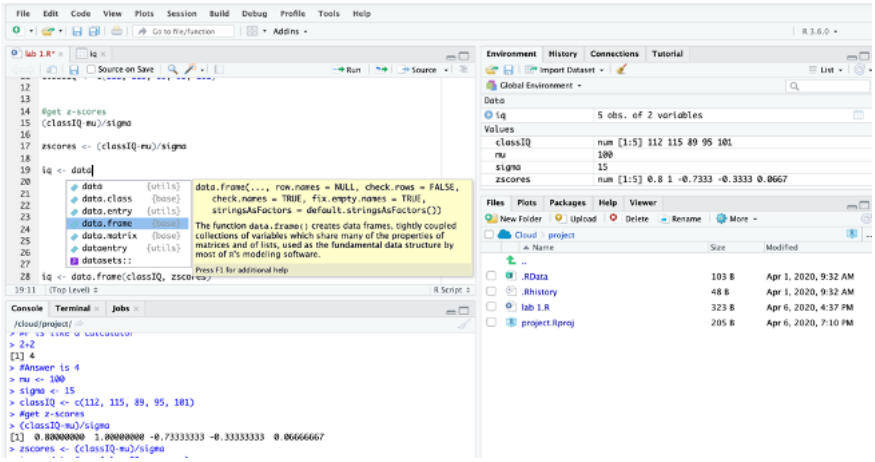
Double-click on the word 'iq' in the environment panel to look at the dataset that you just created (it is circled in red in the picture above).

A new tab will open with a spreadsheet view of the dataset. When you are done viewing the data, you can close it by click on the 'x' next to the name iq.



Note that after looking at the dataset this way, the command `view(iq)` appeared in the console. You can look at the dataset with the **view** command as well

Finally, when typing the code to create the data frame, you may have noticed that RStudio uses **predictive text**. This means that RStudio will suggest functions and objects as you type. You should take advantage of this nice feature!



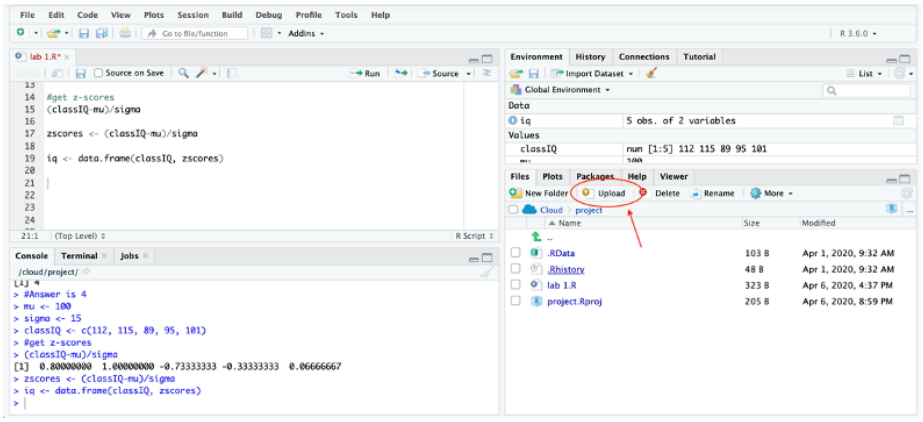
3.2 Importing data into Rstudio cloud

In the Introduction section you learned how to assign data to a **vector** using the **combine** function.

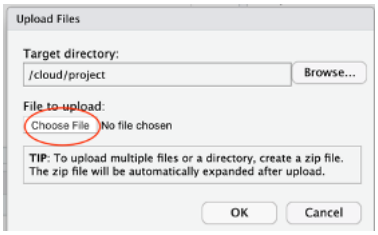
Another way to assign data to an object is by first entering the data into a spreadsheet (like google sheets or excel) and then import the data into RStudio. This will be our preferred method.

First download the exam2.csv file from d2l.

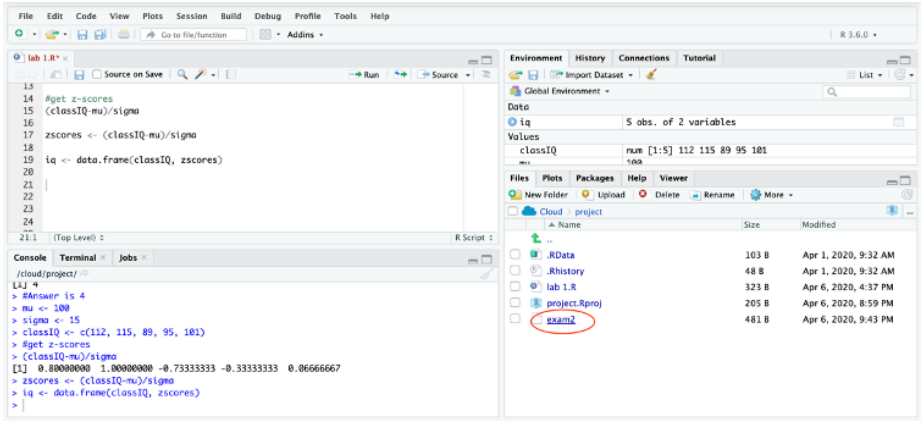
Then select the UPLOAD button in the files window.



In the Upload Files window, click the CHOOSE FILE button and then navigate to the exam2.csv file on your computer. Then click the OK button.

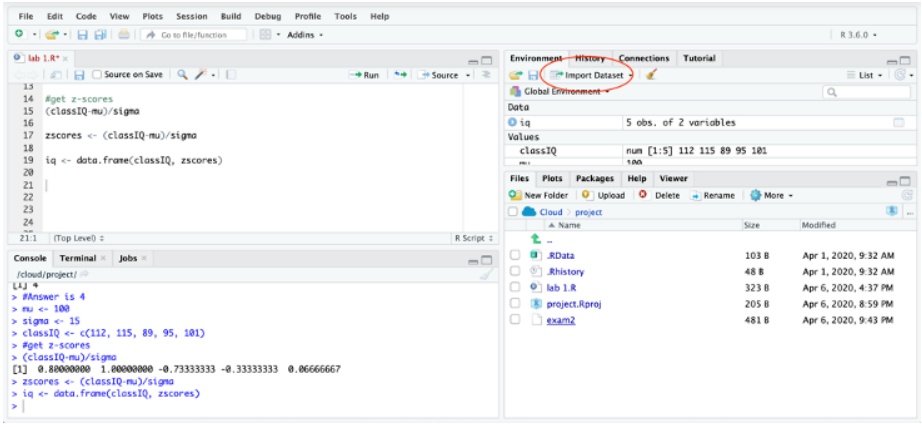


The data file should now be listed in the files section of RStudio.

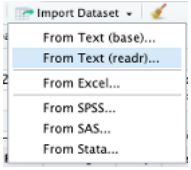


Then you need to import the data into the environment (i.e. assign the data to an object). This can be done through using point and click options or with code.

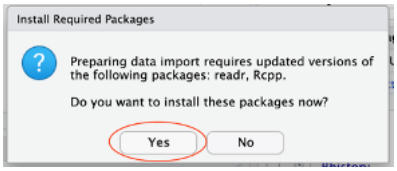
For point and click: First click on the **IMPORT DATASET** button in the environment panel.



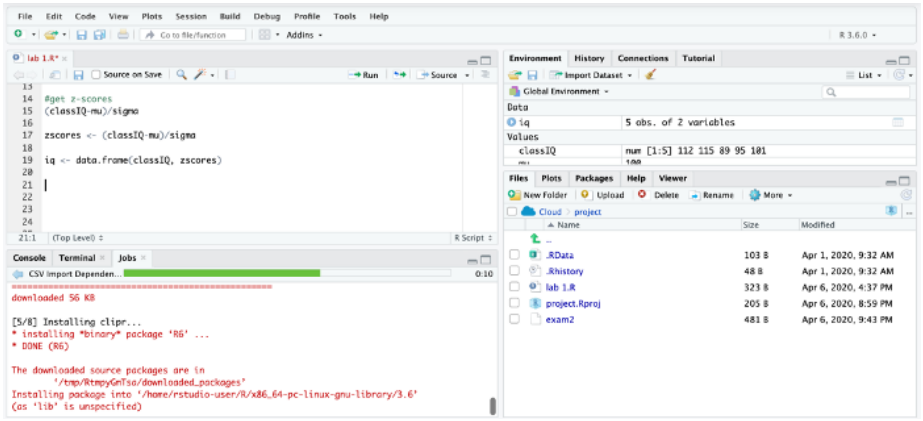
Then select the **'FROM TEXT (READR)'**



The first time you select this – the following window will appear asking if you would like to install the readr package. Select YES. I will introduce packages in the next section.

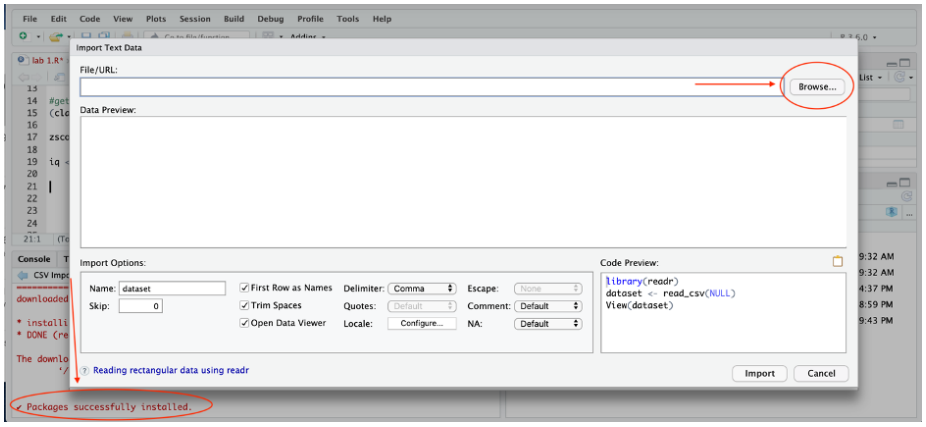


After you select yes, R will begin downloading the package. This can take a few minutes and will look something like this:



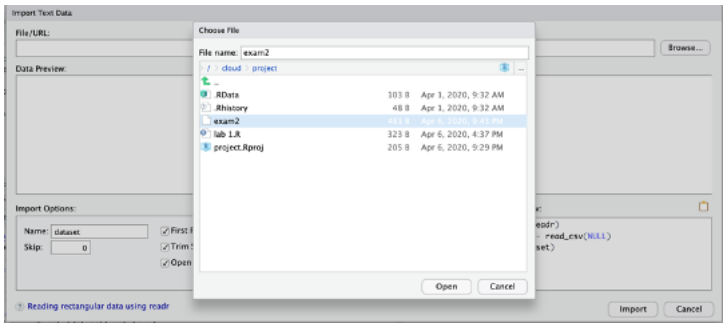
It is important to be *patient* here and let the package download completely before you move on to the next step.

When the download is complete, your screen should look like this:



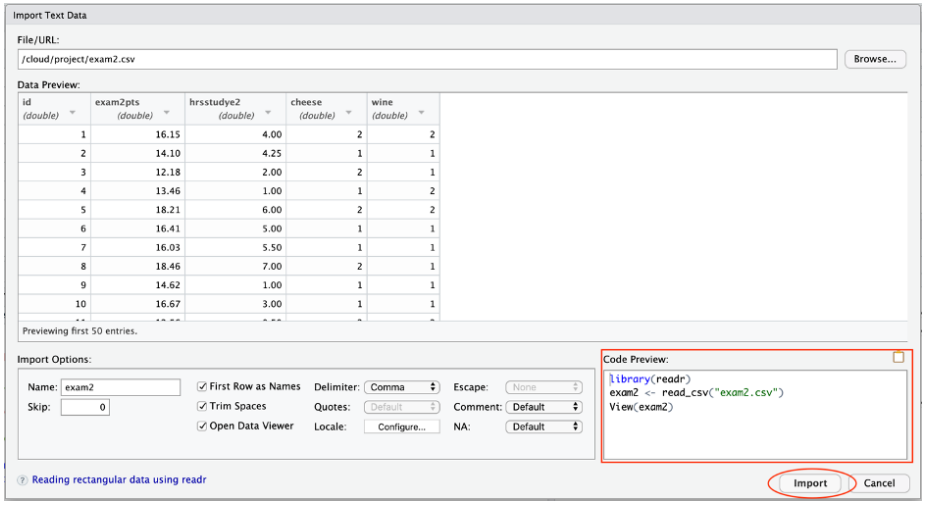
Note that you can see that the package was successfully installed in the console in the bottom left. The next time you use readr to import data – you will not have to download the package first.

Next select the BROWSE button in the top left corner of the import data window.



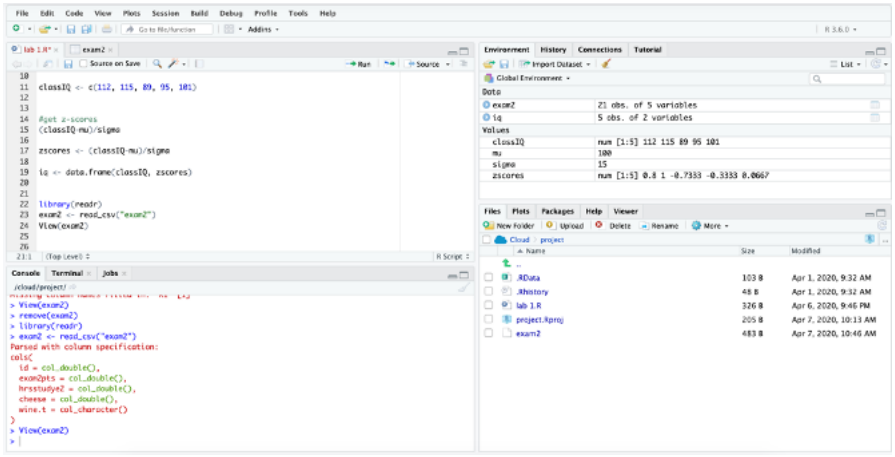
In the choose file window, select 'exam2'. And then select OPEN.

The next window should look like this:

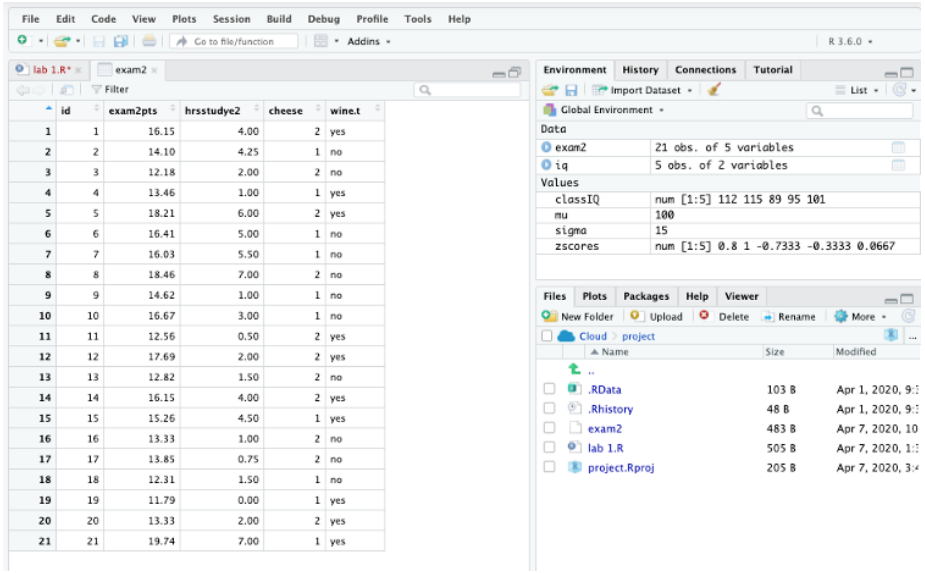


From here you should click the IMPORT button.

But first note the **Code Preview** box. This is the code you could use to import data (instead of clicking through all these windows). Copy this code before I click the import button and then paste it into your script for your records and in case you need to assign the file to an object again (because it is faster with code).



Double click on the word exam2 in the environment panel to look at the dataset.



	id	exam2pts	hrsstudye2	cheese	wine.t
1	1	16.15	4.00	2	yes
2	2	14.10	4.25	1	no
3	3	12.18	2.00	2	no
4	4	13.46	1.00	1	yes
5	5	18.21	6.00	2	yes
6	6	16.41	5.00	1	no
7	7	16.03	5.50	1	no
8	8	18.46	7.00	2	no
9	9	14.62	1.00	1	no
10	10	16.67	3.00	1	no
11	11	12.56	0.50	2	yes
12	12	17.69	2.00	2	yes
13	13	12.82	1.50	2	no
14	14	16.15	4.00	2	yes
15	15	15.26	4.50	1	yes
16	16	13.33	1.00	2	no
17	17	13.85	0.75	2	no
18	18	12.31	1.50	1	no
19	19	11.79	0.00	1	yes
20	20	13.33	2.00	2	yes
21	21	19.74	7.00	1	yes

Each column is a different variable. Each row is a different participant (in this example a student).
- The first column is an arbitrary student ID number – so that the students' identity is protected.
- The second column is exam points earned by the students out of 20 (this is real data from a Fall 2019 class).
- The third column is the number of hours the students studied for the exam (this is made up data).
- The fourth column is whether or not students ate cheese the night before the exam (1 = no; 2 = yes... also made up data).
- The last column is data on whether or not students drank wine the night

before the exam (1 = no; 2 = yes... also made up data).

Chapter 4

Packages

4.1 Installing packages

4.2 Loading Packages

Chapter 5

Picturing Data

Chapter 6

Descriptive Statistics

A **function** in R is any kind of operation. For example, the `mean()` function will compute an average ($sumX/N$).

An *argument* is what a function acts on. The `mean()` function takes one argument, a numeric vector.

For example, `mean(classIQ)` will return the average of the IQ scores in the classIQ vector. This code applies the function mean to the variable classIQ.

This section focuses on functions that find descriptive statistics. Descriptive statistics refer to measures of central tendency (mean, median, and mode) and measures of variability (standard deviation, variance, range, etc.). There are several functions that find descriptive statistics within R. My preferred method uses the Tidyverse and Psych packages, which I describe first. Next I will show you how to find descriptive statistics using base R.

Chapter 7

Measurement

7.0.1 Reliability

7.0.2 Validity

Chapter 8

Basic Data Transformations

Chapter 9

Bivariate correlational research

9.1 Association claim with two quantitative variables

9.1.0.1 Open data

9.1.0.2 Get to know data

9.1.0.3 Test assumptions

9.1.0.4 Compute CI, effect size, and NHST

9.1.0.5 Write up results

Chapter 10

Multivariate correlational research

Chapter 11

Simple experiment

11.1 Independent group design

11.1.1 Two groups

11.1.2 More than two groups

11.2 Dependent group design

11.2.1 Two groups

11.2.2 More than two groups

Chapter 12

Experiments with more than one IV

12.1 Independent-group factorial design

12.2 Within-group factorial design

12.3 Mixed factorial design

Chapter 13

Final Words